Bursty Tracing: A Framework for Low-Overhead Temporal Profiling

Martin Hirzel
hirzel@colorado.edu

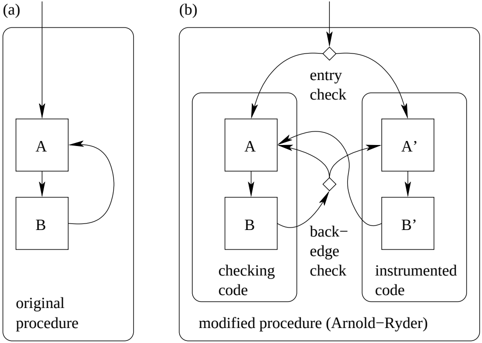
Trishul Chilimbi
trishulc@microsoft.com

FDDO4 December 2001 Austin, Texas

“Low-overhead temporal profiling”

- Low overhead
  - Intended for dynamic optimization systems
  - Profile overhead must be recovered by optimization
- Temporal profiling
  - Trend in profiling literature: discover more causality (path profiling, calling context trees, *etc.*)
  - Temporal profiles expose more optimization opportunities

Arnold-Ryder profiling framework



- Counter *nCheck*
- Sampling rate $r = \dfrac{1}{nCheck_0 + 1}$
- Implemented in Jikes RVM (Java on PowerPC)

Why longer bursts

- Arnold-Ryder framework isolates events by loop back-edges, calls, and returns

- Example:
  for$(i = 1;\ i < n;\ i{+}{+})$
        if$(\ldots)\ f();$
        else $g();$

- Temporal relationships interesting for optimization:

  - Single-entry multiple-exit regions

  - Field reordering

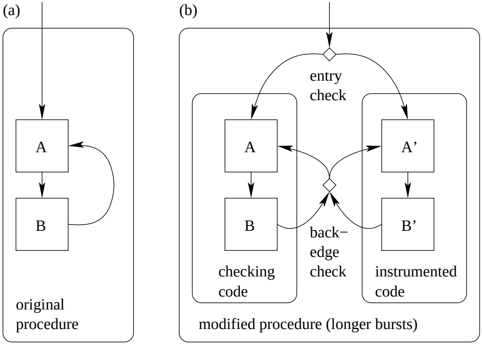
Contributions

- Longer bursts
  - Our framework captures temporal relationships across loop back-edges, calls, and returns.
- x86 binaries
  - We report experiences with the framework in an alternative setting with different advantages and disadvantages.
- Overhead reduction techniques
  - We eliminate some of the checks at procedure entries and at loop back-edges.

Talk outline

- Introduction
- Methodology
  - Longer bursts
  - Overhead reduction by eliminating checks
- Evaluation
  - Overhead
  - Profile quality
- Conclusion

Longer bursts

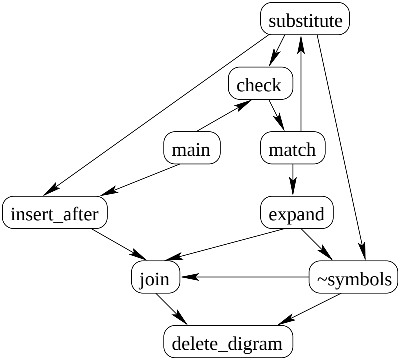


- Counters *nCheck* and *nInstr*
- Sampling rate $r = \dfrac{nInstr_0}{nCheck_0 + nInstr_0}$
- Implemented using Vulcan (x86 binaries)

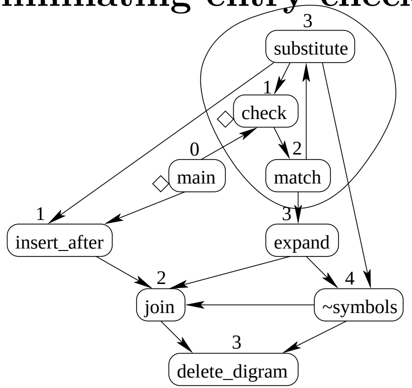
Fewer checks

- Goal: reduce overhead
- Starting point: 6-35% overhead in our setting with checks on all procedure entries and loop back-edges
- Constraint: never recurse or loop for unbounded amount of time without check
- Remark: analogous to thread-yield points, gc-safe points, asynchronous-exception points

Eliminating entry checks



Eliminating entry checks



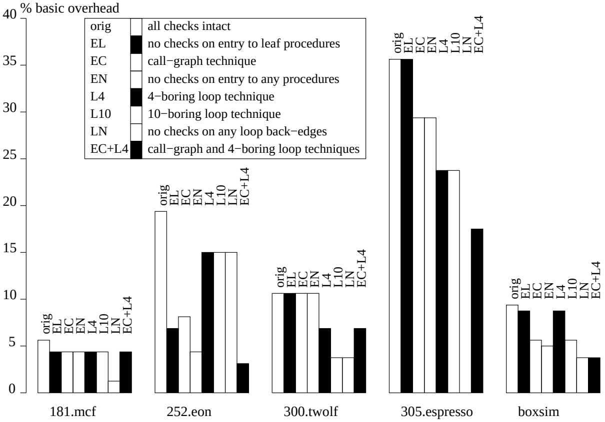
$$C = \Big\{ f \in N \mid \neg \mathit{is\_leaf}(f) \wedge (\mathit{is\_root}(f) \vee \mathit{addr\_taken}(f) \vee \mathit{recursion\_from\_below}(f)) \Big\}$$

Eliminating loop back-edge checks

- Tight inner loops
  - Checking gets expensive relative to time spent in original code
  - Statically optimized, not much opportunity for dynamic optimization
- Omit both checking and profiling for tight inner loops
- k-boring loop:
  - No calls
  - At most k profiling events of interest

Evaluation: Overhead

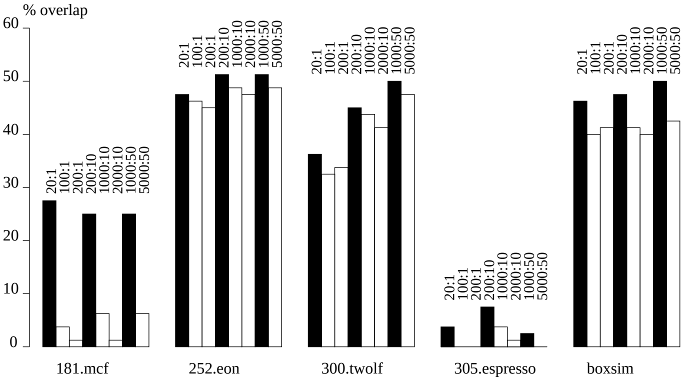
- $overhead(r) = basic\_overhead + r \cdot instr\_overhead$



Case study: Hot data stream profiles

- *data reference*: dynamic load, (*pc*, *addr*) pair
- *data stream*: sequence v of data references
- *heat of data stream*: $v.heat = v.length * v.frequency$
- *hot data stream*: when $v.heat > heat\_threshold$ (we set the threshold such that all hot data streams together cover 90% of the profile)
- *hot data stream profile*: set P of hot data streams and their heats
- $overlap(P, Q) = \sum_{v \in P \cup Q} \min\{v.heat_P, v.heat_Q\}$

Evaluation: Overlap

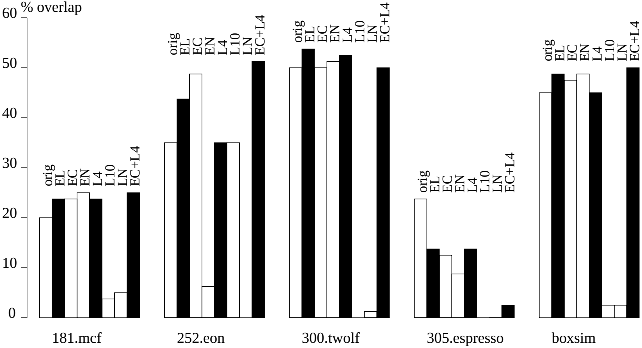


- $nCheck_0{:}nInstr_0$

Evaluation: Overlap

$nCheck_0{:}nInstr_0 = 1000{:}50$

orig	all checks intact
EL	no checks on entry to leaf procedures
EC	call−graph technique
EN	no checks on entry to any procedures
L4	4−boring loop technique
L10	10−boring loop technique
LN	no checks on any loop back−edges
EC+L4	call−graph and 4−boring loop techniques



Related work

- Arnold, Ryder, *A framework for reducing the cost of instrumented code*, PLDI 2001
- Temporal profiling
  - Ball, Larus, *Efficient path profiling*, MICRO 1996
  - Ammons, Ball, Larus, *Exploiting hardware performance counters with flow and context sensitive profiling*, PLDI 1997
  - Larus, *Whole program paths*, PLDI 1999
  - Chilimbi, *Efficient representations and abstractions for quantifying and exploiting data reference locality*, PLDI 2001

Conclusions

- Bursty tracing can collect temporal profiles online
  - General, low-overhead, deterministic
  - Flexible trade-off between sampling rate, overhead, and burst-length
  - Temporal
- Future work
  - Prefetching hot data streams
  - Eliminating more loop back-edge checks
  - Improving profile quality further